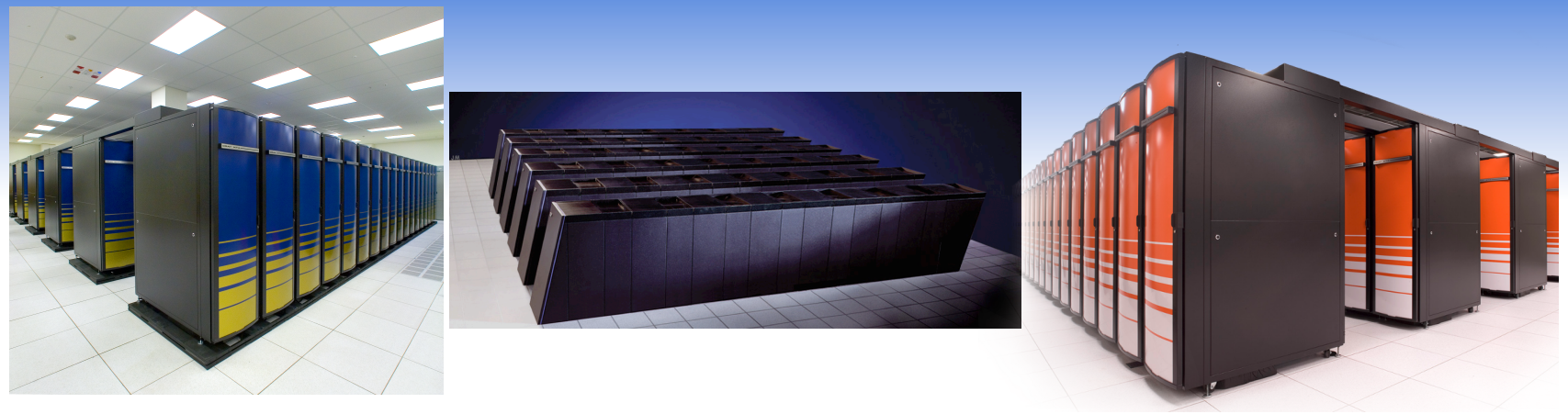


HDF5: Initial Ideas for Performance Tuning

John Shalf

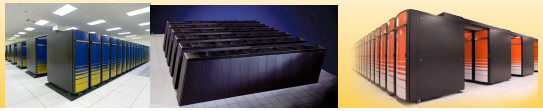
(and Shane, Ruth, and Quincey)

HDF Worshop

January 20, 2009

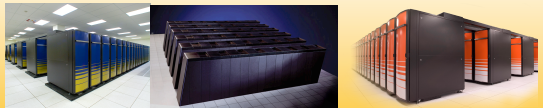
Options for Additional Info on Wednesday

- **Ruth and Quincey:** Any HDF5 tutorial you want
- **Katie** : Lustre Striping and More detail on performance tuning for Lustre
- **John** : HDF5 for native speakers of NetCDF and HDF4
- **Andrew**: Lustre Monitoring FrameworkDavid Knaak: MPI-IO performance tuning for Cray
- **Rob**: More ROMIO and HDF5 tuning
- **David & Noel:** IPM profiling



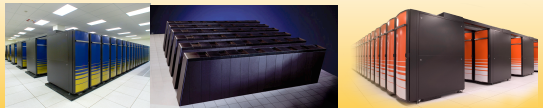
Purpose of Workshop

- **Need to define action plan to tune HDF5 performance**
  - Learn how HDF5 is being used by current applications
  - Understand where you are experiencing performance problems given use cases
  - Understand root cause of performance problems
  - Prioritize plan to fix problems
- **Please provide your personal list of problems that you would like "fixed"**
  - Feel free to do some real-time editing of your ppt



Application I/O Kernels to Measure Progress (*Cannot improve what you don't measure)*

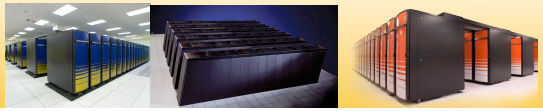
- IOR
- FLASH I/O
- Chombo HDF5
- Vorpal / 3DIO
- NetCDF4 tests (IPCC4, Randall Code)
- Cactus HDF5 Bench
- S3D
- What else?
  - Work together with other procurements to collect tests to get good coverage



Strawman Action Plan

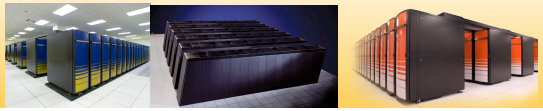
- **Strawman Development Plan to improve HDF5 integration with NERSC platform**
  - Based on NERSC apps experience
  - Not comprehensive
    - does not include your experience/requirements
    - Very NERSC-platform specific

- **Please help us make this more responsive to your needs**



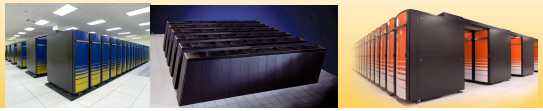
A Few Performance Principles
(some assumptions)

- **Small writes are bad (aggregate to >1MB operations)**
- **Use wide striping on Lustre for parallel I/O**
- **Choose #stripes to be multiple of #clients**
  - Best to set striping before writing to file
- **Use transaction sizes equal to stripe size**
- **Align writes to stripe boundaries**
  - Even if writes to file are sparse
- **2-phase I/O to fix alignment issues**
  - # I/O clients equal to #OSTs assigned
  - Reorganize I/O so that it is always aligned to OSTs (e.g. Data organized so Client #1 always handles transactions for same OST)



Baseline Lustre Integration

- **Issue**: HDF5 has no direct access to Lustre tunable parameters
  - Stripe width, number of stripes, stripe offsets
- **Strategy:** expose lustre tuning interfaces via H5P interface
  - HDF5: has H5P interface for uniform access tunable parameters
  - Lustre: has lustre_user.h that defines lustre-specific ioctls() to introspect and modify tunable parameters
- **Benefits:** first step to tuning HDF for Lustre
  - Enable user I/O libraries to introspect and manipulate HDF interfaces and Lustre to tune for performance
  - Enable HDF5 to auto-tune performance tunables based on Lustre parameters
  - **Autotuning**

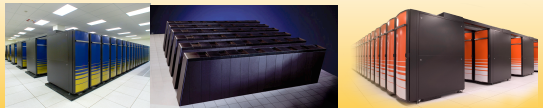


Ergonomic issues: *How do we expose tuning capabilities?*

- **Use Lustre-specific Virtual File Driver**
  - Automatically chooses good values for tunable params without user intervention
- **Define Lustre-specific behavior in mpi-posix and mpi-io VFDs (#ifdef)**
  - User code uses H5P interface to query and set Lustre parameters (and any HDF5 tunable parameters to match)
  - Use new H5P interface to request ***auto-tuning*** of parameters
- **Depend on MPI-IO hints?**

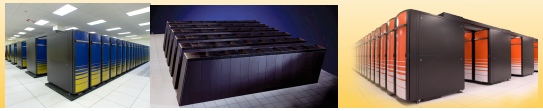
Lustre File Striping

- **Issue**: Users usually forget to stripe file properly for parallel I/O
  - Best performance if striping is an even multiple of number of client processors
  - Correct striping choice usually indicated by how many clients open same file
  - New Lustre 1.6 striping behavior will use same OST twice in striping (need to nail down independent stripes)
- **Strategy**: Use Lustre-specific H5P interfaces to set striping
  - Set striping at file open
  - Have lustre-specific VFD automatically choose good striping based on number of clients
  - Have user request auto-tuning through H5P "auto-tuning" API call



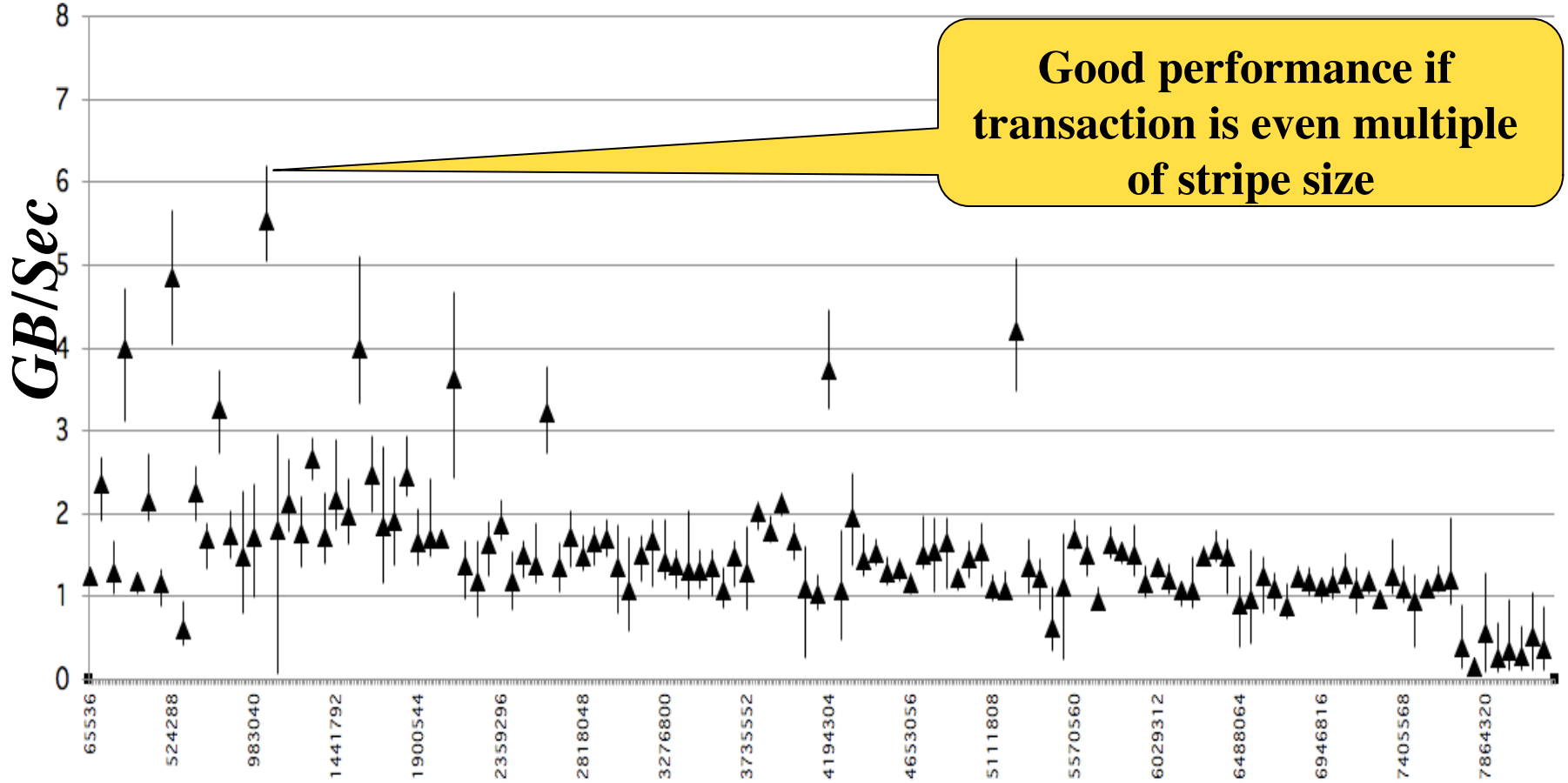
HDF5 Object Alignment

- **Issue**: Lustre hates unaligned data
  - HDF5 inserts various-sized data objects into file to conserve space by default
  - Lustre performs poorly for unaligned writes
- **Strategy**: Use Lustre-specific ioctls() to set correct HDF5 tunable parameters for alignment
  - Make HDF5 prefer stripe-sized objects
  - HDF5 H5P_setalign() tunable parameter allows objects to automatically be aligned to Lustre stripe boundaries
  - Lustre ioctls() enable HDF5 to find optimal alignment for objects (stripe boundaries)
- **Benefits**:
  - Automatically set optimal alignment for HDF5 objects



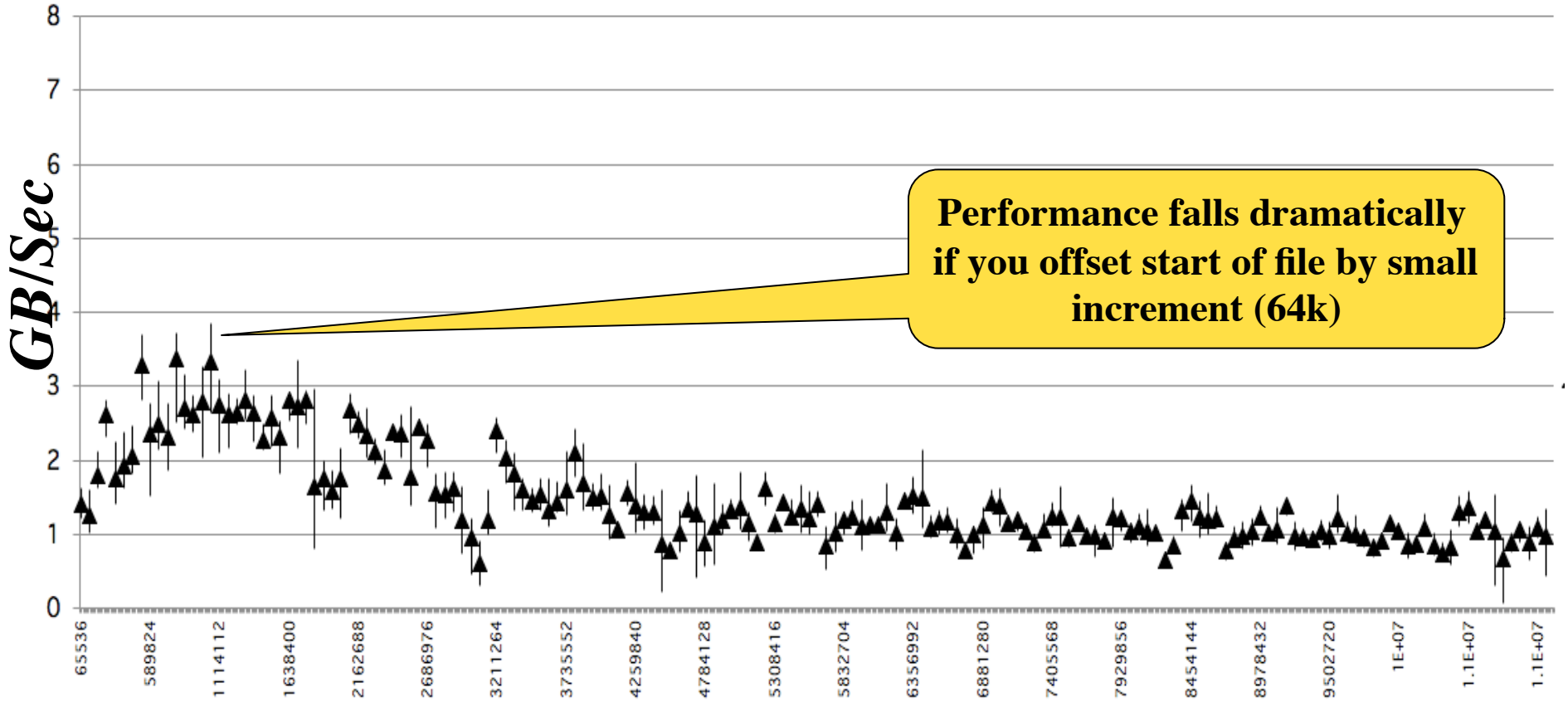
I/O Performance Sensitivity to Transfer Size

2GB File Size, 80 Processors, 40 OSTs



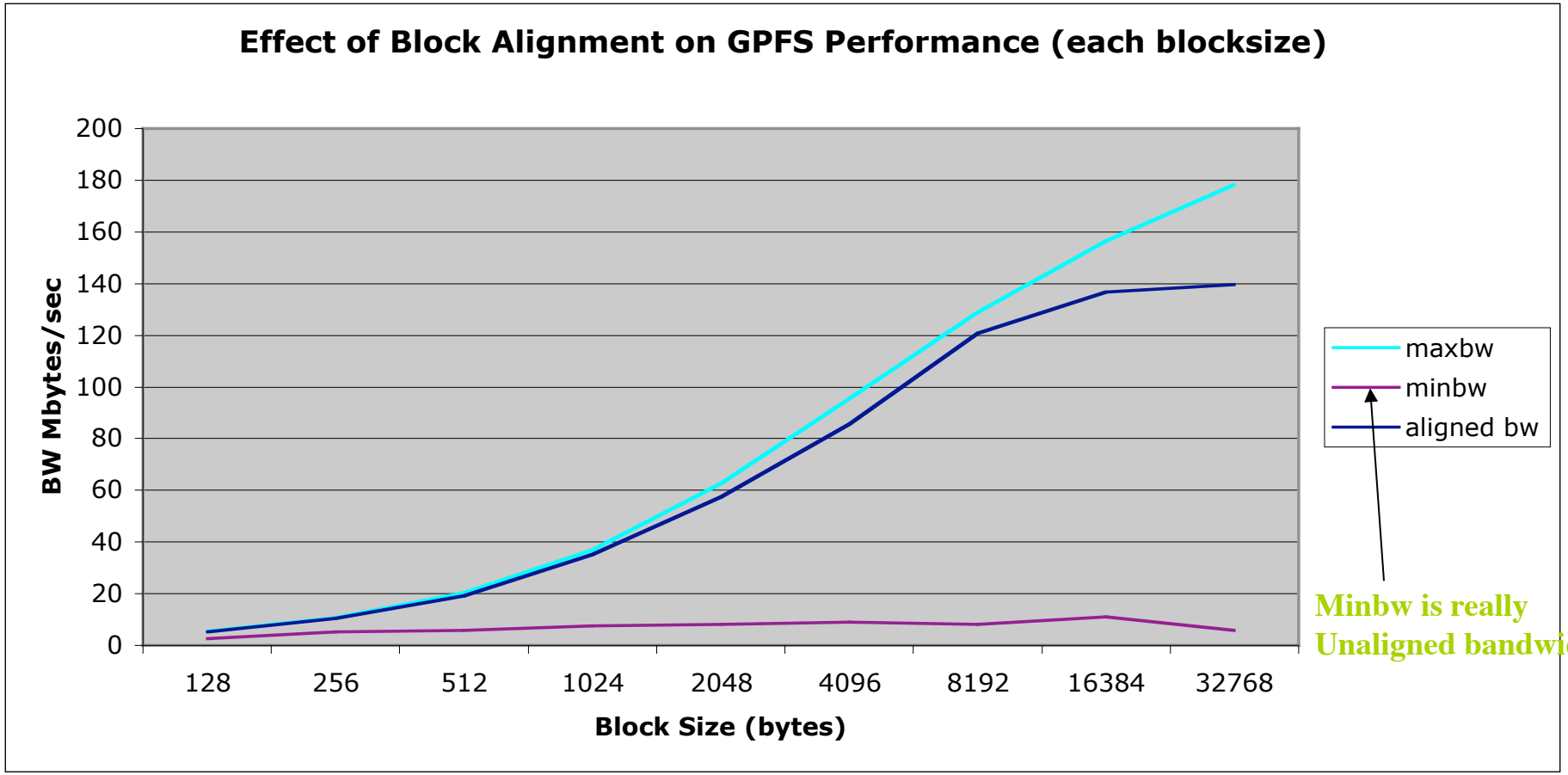
I/O Performance Sensitivity to Transfer Size

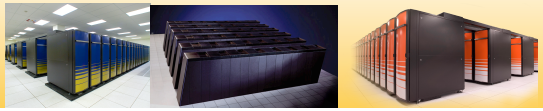
2GB File Size, 80 Processors 40 OSTs: Offset file start by 64k



Streaming Unaligned Accesses

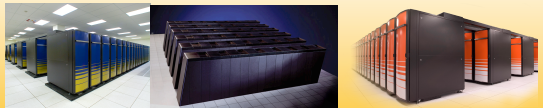
(not to pick on Lustre… GPFS suffers too)





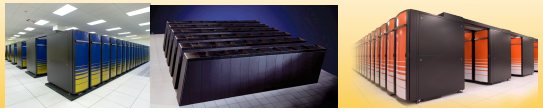
HDF5 Object Alignment

- **Issue**: Lustre hates unaligned data
  - HDF5 inserts various-sized data objects into file to conserve space by default
  - Lustre performs poorly for unaligned writes
- **Strategy**: Use Lustre-specific ioctls() to set correct HDF5 tunable parameters for alignment
  - Make HDF5 prefer stripe-sized objects
  - HDF5 H5P_setalign() tunable parameter allows objects to automatically be aligned to Lustre stripe boundaries
  - Lustre ioctls() enable HDF5 to find optimal alignment for objects (auto-tune for stripe boundaries)
- **Benefits**:
  - Automatically set optimal alignment for HDF5 objects

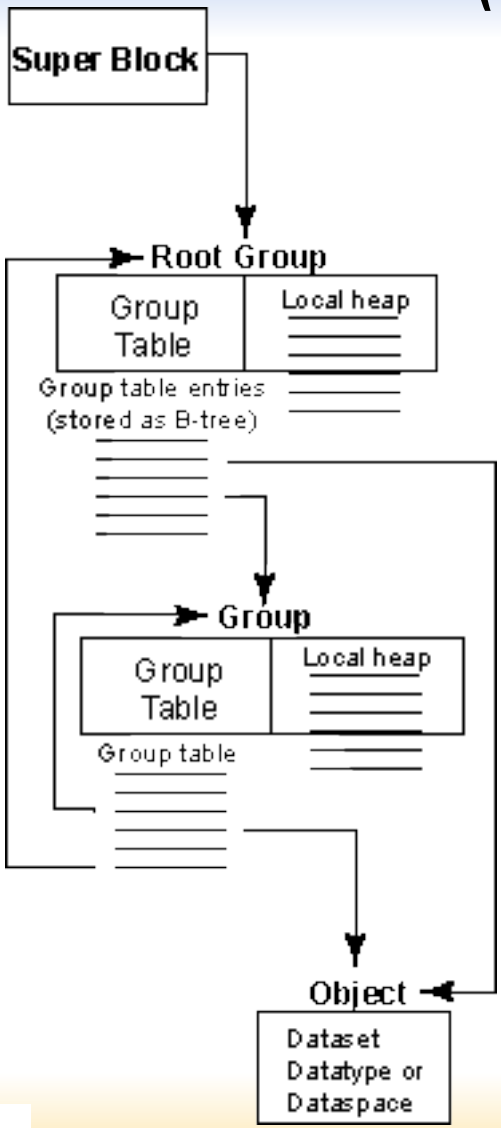


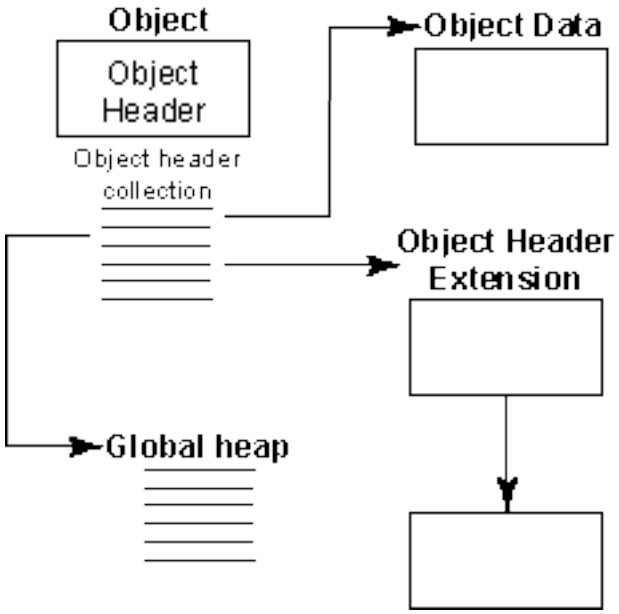
HDF5 Metadata and Index block Tuning

- **Issue**: HDF5 hierarchical indices and metadata blocks
  - HDF5 uses hierarchical indexing scheme and compact metadata to conserve space by default
  - High-bandwidth I/O systems perform badly for small transactions (metadata cache helps, but physical layout can also be modified to favor aggregation of indices)
- **Strategy**: Lustre introspection to set
  - Use Lustre ioctls() to find stripe size and stripe boundaries
  - Set HDF5 tunable parameter for size of indexing blocks make it equal to Lustre stripe size
  - Also set HDF5 tunable for metadata cache size to accommodate caching that is multiple of stripe size

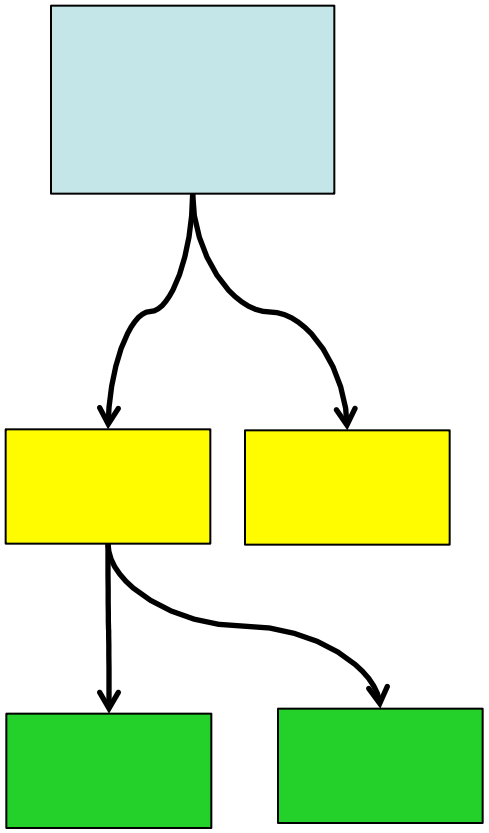


HDF5 Index Blocks
(B-tree indices)

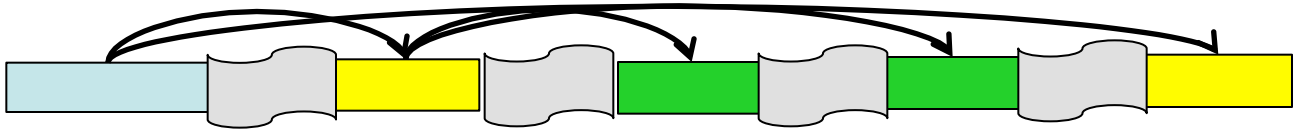




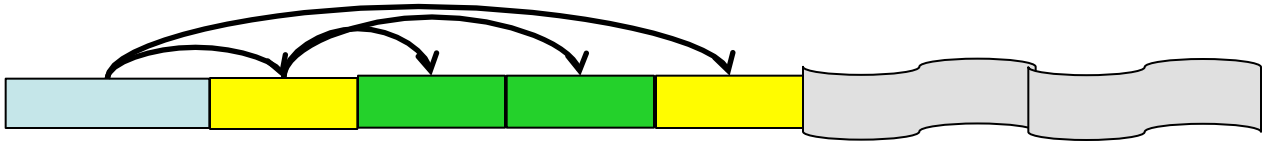
HDF5 Index Blocks



Flattened Represnetation



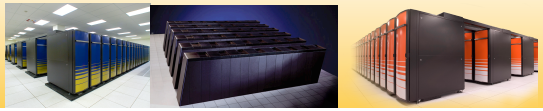
B-Tree index objects distributed throughout file
- **Not-multiples of stripe-size**
- **Not stripe-aligned**



Target larger (1MB sized)
Pad blocks to align them to stripe boundaries

HDF5 Metadata and Index block Tuning

- **Issue**: HDF5 hierarchical indices and metadata blocks
  - HDF5 uses hierarchical indexing scheme and compact metadata to conserve space by default
  - High-bandwidth I/O systems perform badly for small transactions (metadata cache helps, but physical layout can also be modified to favor aggregation of indices)
- **Strategy**: Lustre introspection to set
  - Use Lustre ioctls() to find stripe size and stripe boundaries
  - Set HDF5 tunable parameter for size of indexing blocks make it equal to Lustre stripe size
  - Also set HDF5 tunable for metadata cache size to accommodate caching that is multiple of stripe size



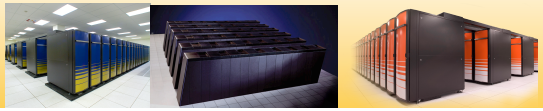
Other HDF5 Metdata Optimizations

- **Issue**: When HDF5 reads parallel file makes redundant requests for same metadata block at each client
- **Strategy**: Read metadata from master and broadcast to clients

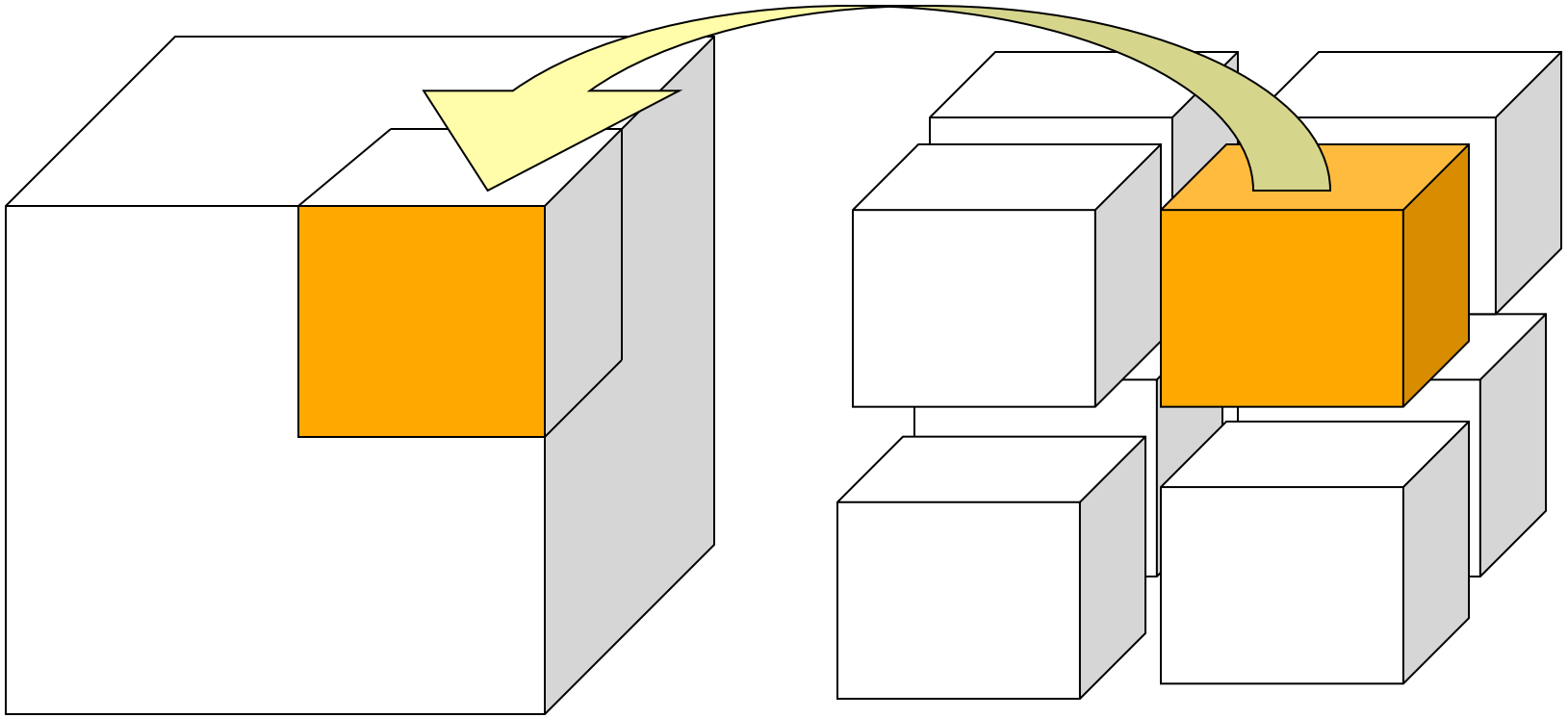
- **Issue**: Processor "0" handles metadata
  - Can create load-imbalance (Amdahl's law hurts)
- **Strategy**: Use POSIX async (or some other async) method to hide metadata operations
  - POSIX async I/O interface
  - Dedicated "metadata" server process

Parallel I/O: Strided Data Access

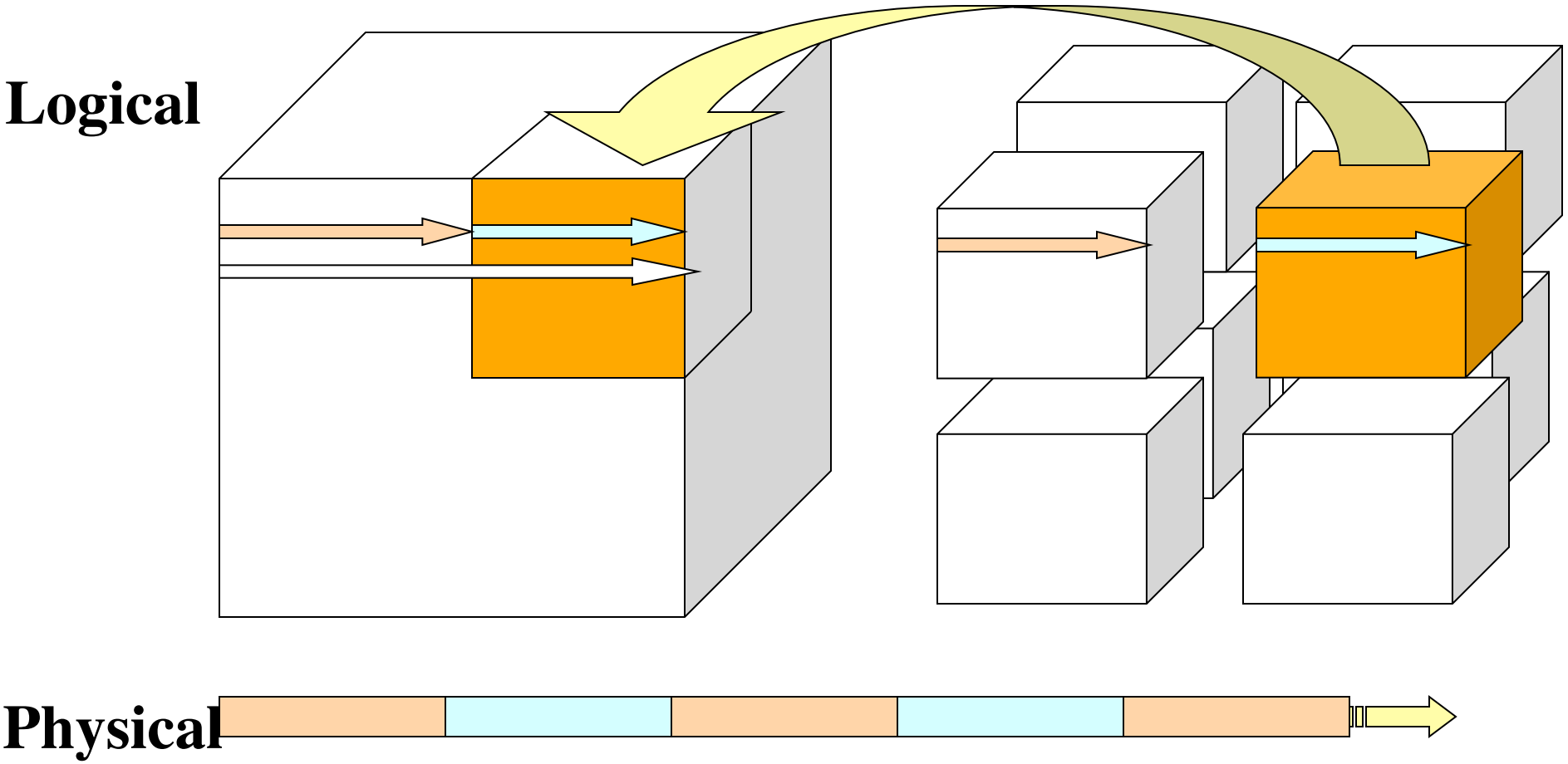
- **Issue**: Parallel I/O to shared file requires fine-grained strided data access patterns (undo domain decomp)
  - perform poorly due to lock manager and small transaction sizes
  - HDF5 not passing sufficient information to lower layers of I/O stack (MPI-IO or filesystem tuning APIs)
- **Strategy**: Describe intended pattern using MPI-IO hints or filesystem-specific hints
  - Already plenty of code in HDF5 to use GPFS hints interfaces and data structures
  - How do we provide such detailed hints to Lustre?
  - Can we provide better hints to MPI-IO in the MPI-IO VFD



3D (reversing the domain decomp)

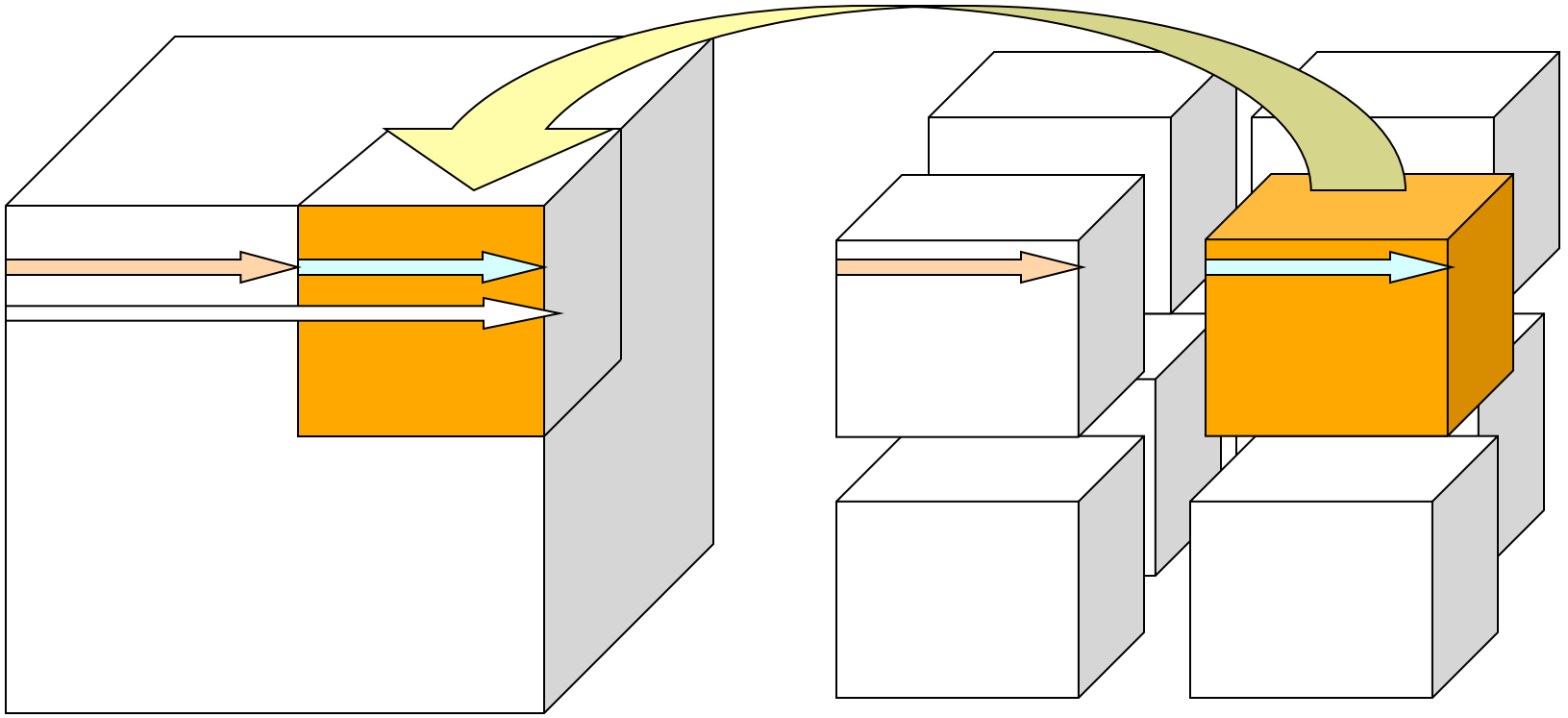


3D (reversing the decomp)

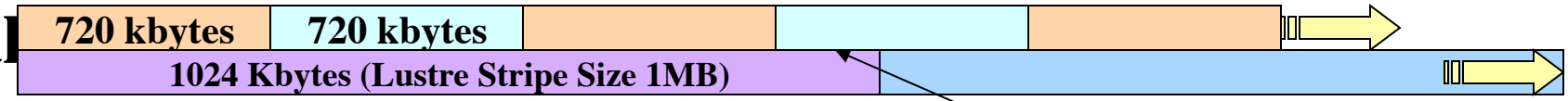


3D (stripe alignment issues)

Logical



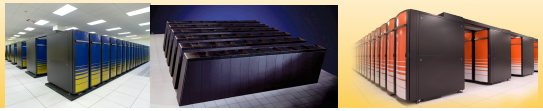
Physical



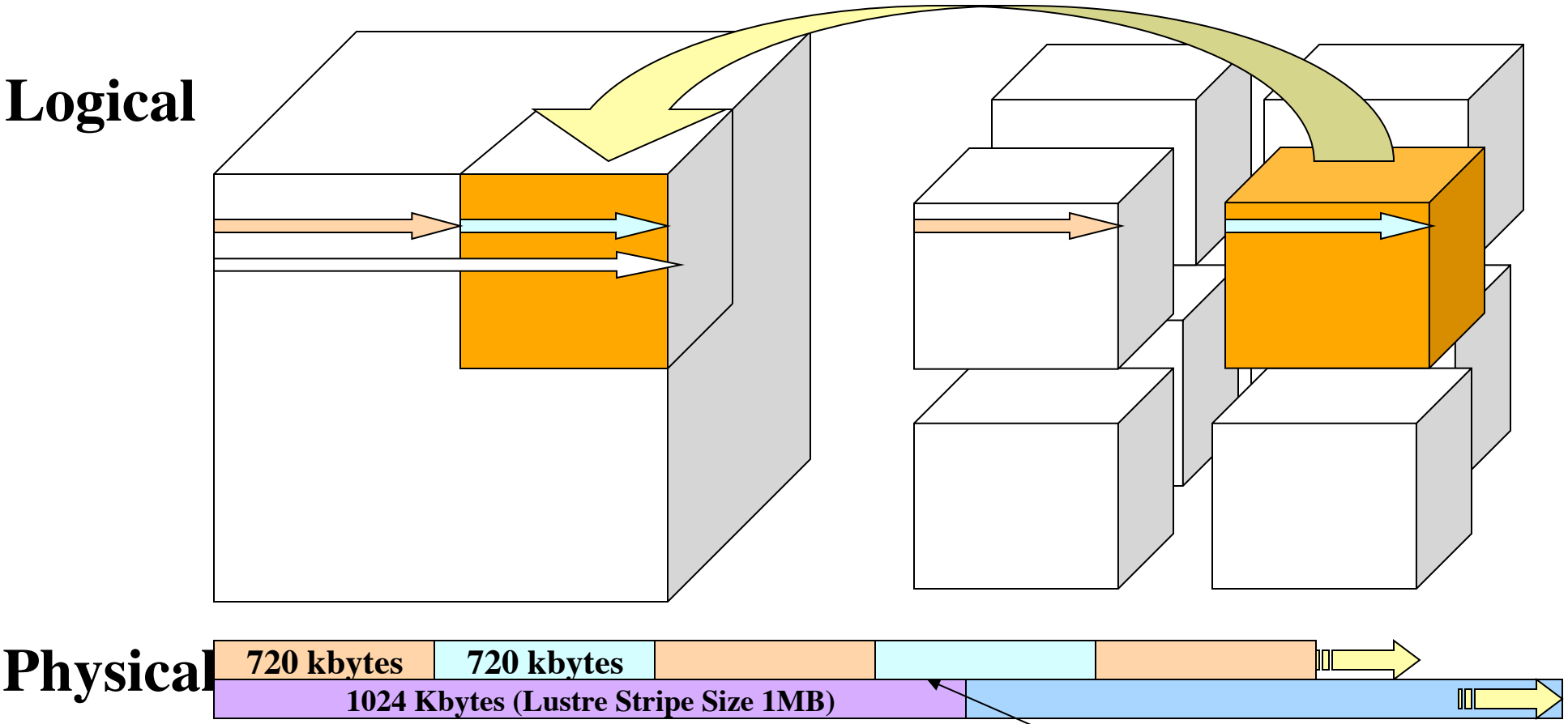
Writes not aligned to block boundaries

Parallel I/O: Strided Data Access

- **Issue**: Parallel I/O to shared file requires fine-grained strided data access patterns (undo domain decomp)
  - perform poorly due to lock manager and small transaction sizes
  - HDF5 not passing sufficient information to lower layers of I/O stack (MPI-IO or filesystem tuning APIs)
- **Strategy**: Describe intended pattern using MPI-IO hints or filesystem-specific hints
  - Already plenty of code in HDF5 to use GPFS hints interfaces and data structures
  - How do we provide such detailed hints to Lustre?
  - Can we provide better hints to MPI-IO in the MPI-IO VFD

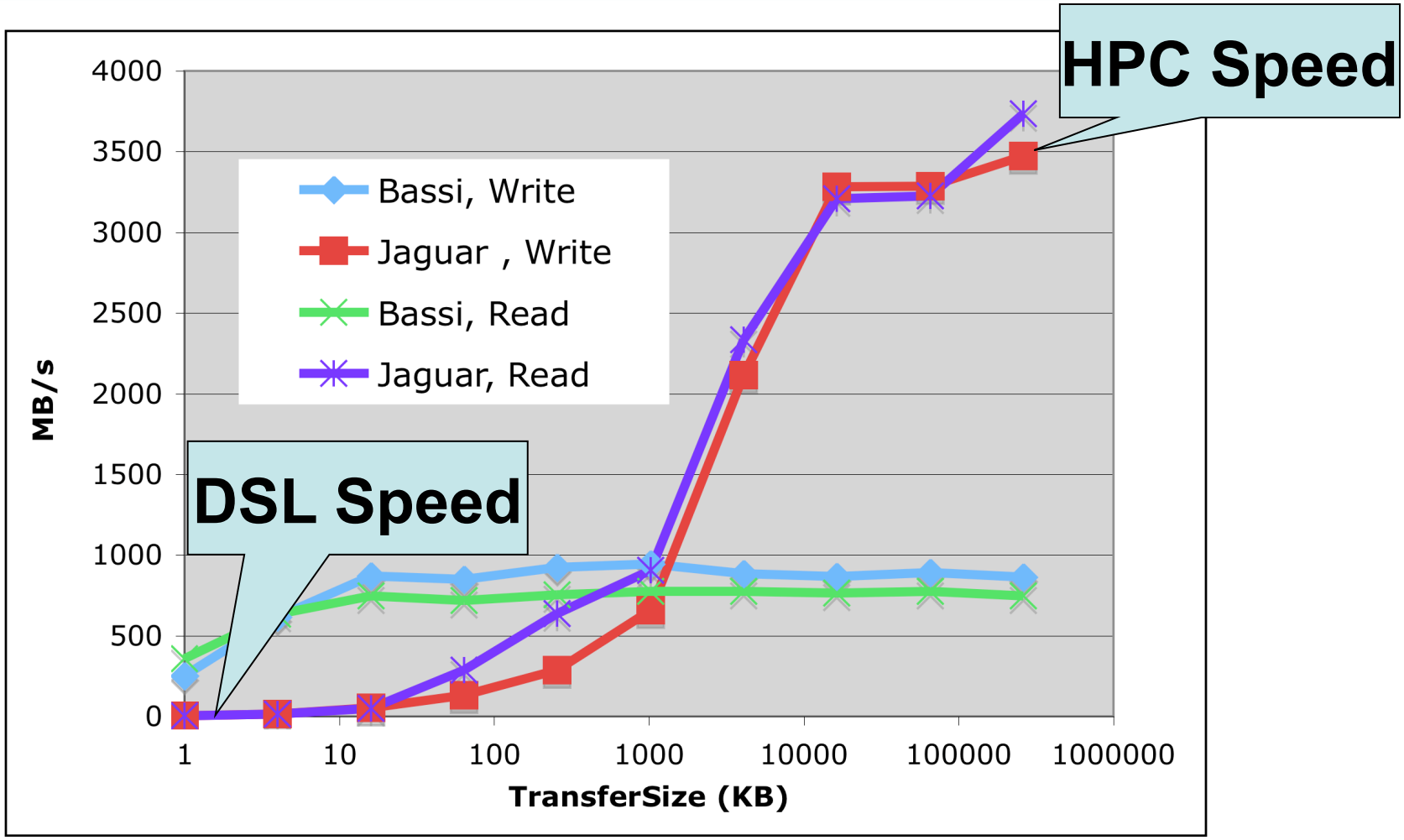


3D (stripe alignment issues)



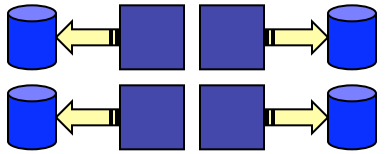
Amdahl's Law Hurts

(aggregate small metadata operations)

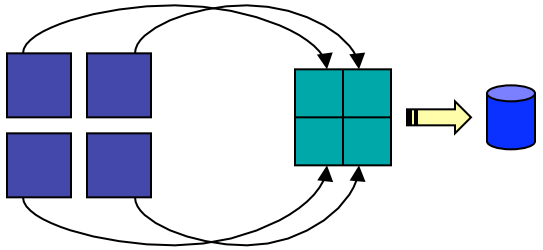


Common Physical Layouts For Parallel I/O

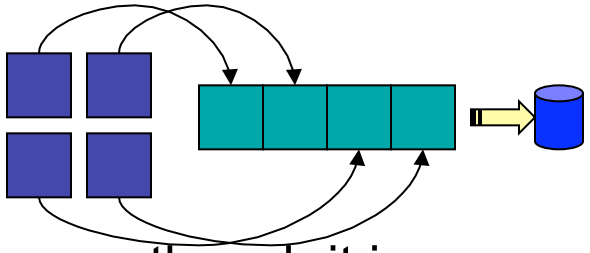
- One File Per Process
  - Terrible for HPSS!
  - Difficult to manage



- Parallel I/O into a single file
  - Raw MPI-IO
  - pHDF5 pNetCDF



- Chunking into a single file
  - Saves cost of reorganizing data
  - Depend on API to hide physical layout
  - (eg. expose user to logically contiguous array even though it is stored physically as domain-decomposed chunks)



HDF5 Chunking

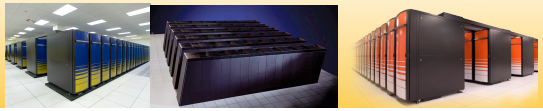
- **Issue**: Performance problems with non-uniform chunking
  - Fully populated HDF chunks go to disk in a single write
  - Partially populated chunks are written using sequence of small/strided writes
- **Strategy**: Write image of partially populated chunk in-memory, then commit to disk
  - Initial experiments by Shan show some performance benefits
  - *Still an issue for non-uniform chunking (defining non-uniform chunk sizes)*
  - *Not all domain decompositions are regular (what to do?)*

HDF5 Chunking2 (2-phase I/O)

- Issue: Chunking oriented towards creating contiguous objects based on the application's domain decomposition
  - Chunksize == local subdomain size
  - However, Lustre experience indicates preference for chunks that are stripe-aligned and stripe-sized
- Strategy: Two phase I/O to subset of writers
  - Define subset of clients as I/O aggregators (ROMIO)
  - Reorganize data via MPI messages to I/O servers from other clients to be stripe-aligned (and clients always hit same OST)
- Options
  - Make it work correctly in MPI-IO?
  - Implement it ourselves in MPI-POSIX VFD
    - Server-directed I/O (PANDA) approach
  - Doing it in user-space is really hard

Other Stuff

- HDF5 Profiling layer
  - Quincey: history comparisons to see if property change had an effect on performance
- HDF5 restricted API
- Auto-tuning
- Going straight from description of datamodel (XML perhaps) to veneer API



We Want Your Input!

- This is not the final action plan
  - Just initial straw-man example of what we can target
  - We want your input
  - Action plan will change to incorporate your ideas, use-cases and experience

- Please tell us what your priorities and performance pain-points are